1

Introducing the Project: The SmartCA Application

COPYRIGHTED MATERIAL

The project for this book is based on a real application for a real company. The names of the company and the application have been changed for privacy reasons. The fictional company name will be Smart Design, and the name of their new application will be called SmartCA. Smart Design is a growing architectural, engineering, and interior design firm. One of its many service offerings is construction administration, which in its case consists mostly of document management, cost control, and project portfolio management.

The Problem

To manage its construction administration (CA) data and processes, Smart Design has been getting by for 10 years on a home-grown Microsoft Access database application, called the Construction Administration Database, which lives on its corporate network. The company has grown accustomed to this application, both the good parts and the bad. When the application was originally written, there were only a few users, the requirements were very simple, they already had licenses for Microsoft Office, and they had a very small budget. All of this made using Microsoft Access a good technology choice. Figure 1.1 shows the main screen of the application.

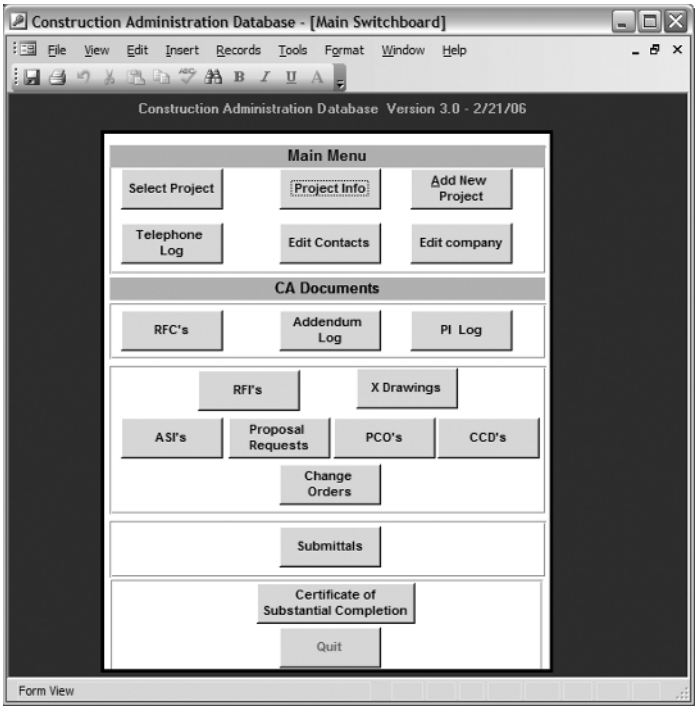

Figure 1.1: Legacy Construction Administration application main screen.

As the years went by, the application became more and more important. It was modified many times, both with code and tweaks to the design of the user interface (UI). This led to UI forms with lots of logic embedded in them as well as some embedded logic in the database queries. The application is now, essentially, a poster child for the Smart Client anti-pattern.

> **The Smart Client anti-pattern is defined by Eric Evans as "Put all the business logic into the user interface. Chop the application into small functions and implement them as separate user interfaces, embedding the business rules into them. Use a relational database as a shared repository of the data. Use the most automated UI building and visual programming tools available" (Evans, *Domain-Driven Design: Tackling Complexity in the Heart of Software* [Addison-Wesley, 2004], 77).**

Figure 1.2 shows the architecture of the current application.

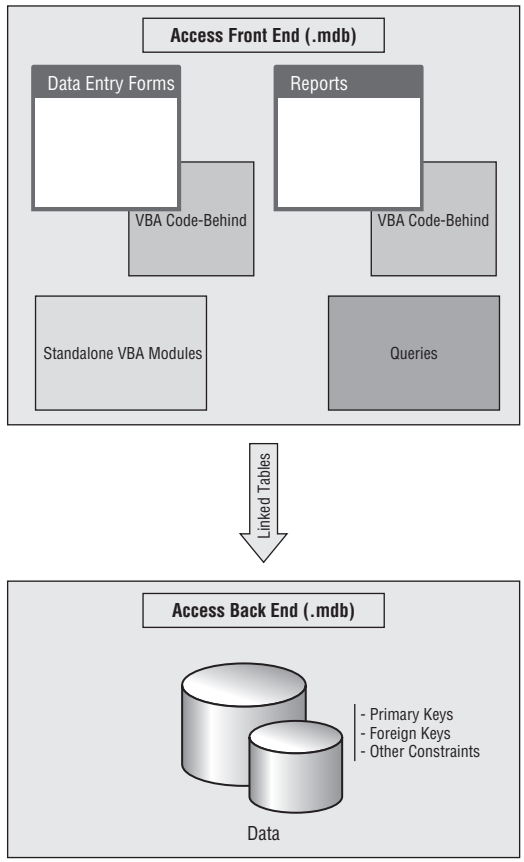

Figure 1.2: Legacy Construction Administration application architecture.

Recently, Smart Design merged with another architectural design company, and as a result the CA application became even more important. It is now being used more often than before by many more users from several remote offices. The increased use has caused scalability and performance problems with the application.

The problem with the Access application is that it has so much logic embedded into its forms, queries, and reports. This makes it very hard to maintain as well as very difficult to add new features to the application.

The Design

As far as a Microsoft Access application goes, the architecture is really not that bad. As shown in Figure 1.2, it is a two-tier application, with all of the UI, business logic, and queries in the first tier (which is a separate `.mdb` file), and all of the database tables in the second tier (also a separate `.mdb` file). Although the current solution for construction administration might have been a good fit 10 years ago, Smart Design has outgrown the application and needs a new solution that will support its new needs. Ten years ago, their needs were fairly simple—they just needed an organized way to capture information and be able to print reports on that information to send out to various people on various projects.

Originally, the main requirement was for a very simple information tracking and reporting tool. The first version of the application was made without any IT involvement, just a stakeholder and one Access programmer. Many changes were made to the program over the years, both by the stakeholder, and by the Access programmer. Several of the changes resulted in denormalized data structures, repetitious code, and various other code smells. Often, changes were made to the application that the Access programmer was not even aware of, and changing things to make them right would have taken a lot of time and effort, so the application just kept on moving along. As the data being tracked started to get larger, an archiving solution was put in place, which resulted in more Microsoft Access data files being created. In the end, almost every attempt to enhance the application has resulted in some type of "one-off" solution that has become very difficult to maintain over the years.

Now that the CA application has been deemed critical to Smart Design's business by their chief operating officer, it has become very apparent that it has greatly outgrown its original design. Smart Design has decided that they do not want to buy an off-the-shelf product; instead they want to rewrite the current application onto a different platform that will meet their growing needs.

Here are their most prevalent needs, in order of importance:

- Reliability and Availability
- Scalability
- Maintainability
- Rich client application functionality
- Offline capable
- Web access
- Intelligent installation and auto-update functionality
- Additional client device support

Reliability and Availability

One of the problems with the current application is that the database sometimes becomes corrupt and must be compacted or repaired regularly, which causes the application to be down for periods of time. The new system's database should be able to be backed up while the system is still in use, and should not be prone to data corruption issues.

Scalability

The system should be able to handle demands from a growing number of users and processes.

Maintainability

Because the code for the application is not correctly partitioned, updating the current application sometimes requires updating the same code and logic in several different places. The new application must be designed it such a way that the domain logic is centralized and is never duplicated.

Rich Client Application Functionality

Users are used to the current Microsoft Access application's rich controls and responsiveness, and they would like to continue to improve upon this type of user experience.

Offline Capable

The new application must be able to work even when the user is not connected to the network. For those users running the application with occasional or intermittent connectivity — such as those used by on-site construction managers, where connectivity cannot be guaranteed at all times — being able to work while disconnected is very important.

Web Access

The firm would like some parts of the application exposed to the web, such as reporting. Also, in the future, some parts of the application may need to be exposed to outside contractors. Another nice-to-have would be the ability to extend the application to support some type of an executive dashboard for showing key performance indicators (KPIs) or similar information to management.

Intelligent Installation and Auto-Update Functionality

Currently, the Smart Design IT department is challenged with making sure that users of the application have the right version on their desktops. IT has also had a tough time getting the application pushed out when new changes have been made to the application. IT would definitely prefer a deployment method similar to that of web applications, and would like SmartCA to be easily installed by clicking on a URL from their intranet. The application must be able to be updated while it is still running, and the updates should guarantee the integrity of the application and its related files.

Additional Client Device Support

The new application should be designed in such a way as to be able to reuse a good part of its core logic modules for different UI devices, such as personal digital assistants (PDAs), smart phones, and the like.

The current application and platform will not easily support these requirements. Therefore, Smart Design has decided to start from scratch and completely reengineer the new application to be able to meet the new requirements. The old Access application has served the company well for more than 10 years. Actually, it can still serve the company well by being the basis for the new design. There are lots

of business rules captured in the old application that are not documented anywhere else, so the old application will be used as a guide in fleshing out some of the requirements for the new system.

The Solution

The new application, SmartCA, will be written using Microsoft Visual Studio 2008 (which includes the Microsoft .NET Framework 3.5) technologies for both the client-side and server-side partitions.

Fulfilling the Reliability, Availability, Scalability, Offline Capable, and Additional Client Device Support Requirements

Most of the current problems in the areas of reliability, availability, and scalability lie in the fact that the legacy application was implemented in Microsoft Access and used Access for its data store. The new solution going forward will be using both a database on the server as well as a database on the client.

On the Server

In order to support the Reliability and Availability requirements, the database server will be a SQL Server instance. All of the data from the legacy application will need to be migrated to the new SQL Server database. A SQL migration script or .NET program will be written that will facilitate this data transfer. This will allow the old application to continue working while the new application is still being built, since the script or migration tool will make it easier to refresh data on a regular basis from the production Access database into the development, testing, and staging database environments. Moving to a server-based relational database (SQL Server) will also lend itself well to the Scalability requirement, although the application design has just as much to do with that as the idea of using a database server instead an Access `.mdb` file for a data store.

On the Client

Yes, that's right, you see it correctly, a database on the client. You are probably saying to yourself, "That is worse than the original Access application's two-tier architecture, where at least the database lived on a network share!" Not so fast, my friend. One of the requirements of the application is to be able to support users who are not always connected to the network, such as those construction managers who may be inside of a construction trailer with no available connectivity, a.k.a. the Offline Capable requirement. The database used on the client will be a SQL Server Compact Edition 3.5 (SQL CE) database. Although SQL CE was originally only targeted for mobile platforms, such as PDAs and Tablet PCs, it now runs on all client platforms. According to Microsoft, SQL CE is a "low maintenance, compact embedded database for single-user client applications for all Windows platforms including tablet PCs, pocket PCs, smart phones and desktops. Just as with SQL Server Mobile, SQL Server Compact is a free, easy-to-use, lightweight, and embeddable version of SQL Server 2005 for developing desktop and mobile applications."

Another benefit of having a database on the client is the fact that it can help take some of the load off the database server, thus helping with the Scalability requirement.

At this point, you may be asking yourself, "Why not use SQL Server Express? At least with SQL Server Express I can use stored procedures!" While it is true that SQL Server Express supports stored procedures, while SQL CE does not, the real reason for using SQL CE is that I want to support multiple devices, not just Windows machines. With SQL CE I can reuse the same database on both a PC and a mobile device, and this functionality maps directly to the Additional Client Device Support requirement. I can live without stored procedures on the client.

Instead of using traditional replication to keep the schema and data between the database on the client and the database server in sync, the application will use Microsoft Synchronization Services for ADO .NET. The Synchronization Services application programming interface (API) provides a set of components to synchronize data between data services and a local store. Equally important is the need to synchronize the local copy of the data with a central server when a network connection is available. The Synchronization Services API, which is modeled after ADO.NET data access APIs, is a much more intelligent, service-based way of synchronizing the data. It makes building applications for occasionally connected environments a logical extension of building applications for which you can count on a consistent network connection. Think about how Microsoft Outlook works, and you will get the picture of the online/offline functionality that the Synchronization Services API will enable.

> *It should be noted that I will not be talking much about databases in this book, since the focus of this book is on Domain-Driven Design. One of the main tenants of Domain-Driven Design is persistence ignorance, and therefore, while the application is being designed, as far as you and I are concerned, the data could be coming from a text file. Therefore, from this point on, I will only talk about the 10,000 foot view when it comes to the database.*

Fulfilling the Maintainability Requirement

In order to avoid embedding business logic in the behavior of the UI elements, such as the various forms, controls, and reports, or even embedded inside of database queries, a layered architecture (ibid., 69) will be used. Because the legacy application was implemented with such a Smart UI anti-pattern, the domain-related code became very difficult to decipher and track down. Unit testing was impossible, and sometimes trying to change one business rule meant tracing of UI code, Visual Basic for Applications (VBA) module code, and embedded SQL code. The layered architecture's main principle is that any element of a layer depends only on other elements in the same layer, or on elements of the layers beneath it. Using a layered architecture will make the code for this application much more maintainable, which maps directly to the Maintainability requirement. The layers that will be used in the SmartCA application will be:

- **UI (presentation layer)** — Probably the easiest to understand, this layer is responsible for showing information to the user and interpreting the user's commands. Sometimes, instead of a human, the user could be another system.
- **Application layer** — This layer is meant to be very thin and is used for coordinating the actions of the domain model objects. It is not supposed to contain business rules or domain knowledge, or even maintain state — that is what the domain model is for. The application layer is very useful for coordinating tasks and delegating actions to the domain model. Although it is not to be used to maintain state of a business entity, it can maintain the state that tracks the current task being performed by the user or system. It is very important that the application layer does not interfere or get in the way of the domain model representing the important parts of the business model (`http://weblogs.asp.net`)

- **Domain layer** — This is where the business logic and rules of an application live, and it is the heart of the software. The domain layer controls and uses the state of a particular business concept or situation, but how it is stored is actually delegated to the infrastructure layer. It is absolutely critical in Domain-Driven Design that the domain layer contains the business model, and that the domain logic is not scattered across any other layers.
- **Infrastructure layer** — This is where general technical, plumbing-related code happens, such as persisting objects to a database, sending messages, logging, and other general cross-cutting concerns. It can also serve as a place for an architectural framework for the pattern of interactions between the four layers. In the next chapter, you will see an example of a framework for the SmartCA domain model that is contained in the infrastructure layer.

Generically, Figure 1.3 shows what the SmartCA layered application architecture looks like.

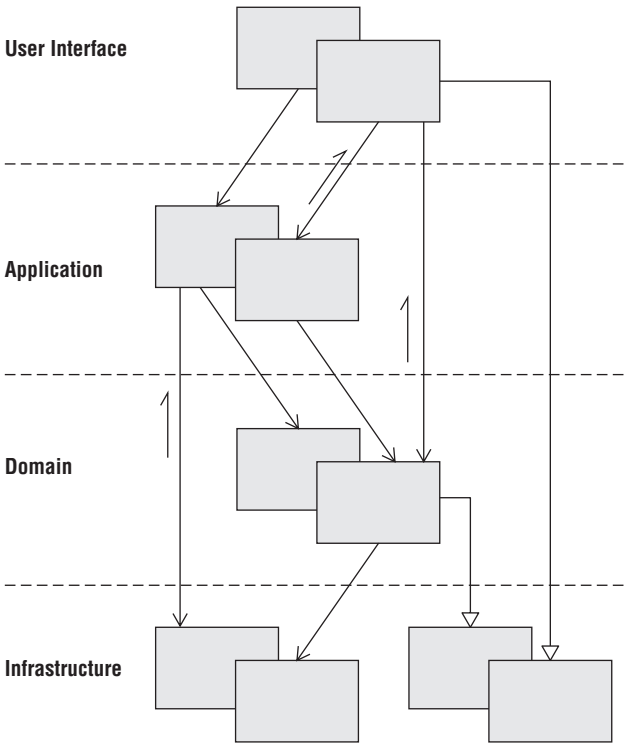

Figure 1.3: The layered architecture (adapted from Evans, 68).

Figure 1.4 shows what the application architecture looks like with all of the technologies and patterns layered on top of the layered architecture model.

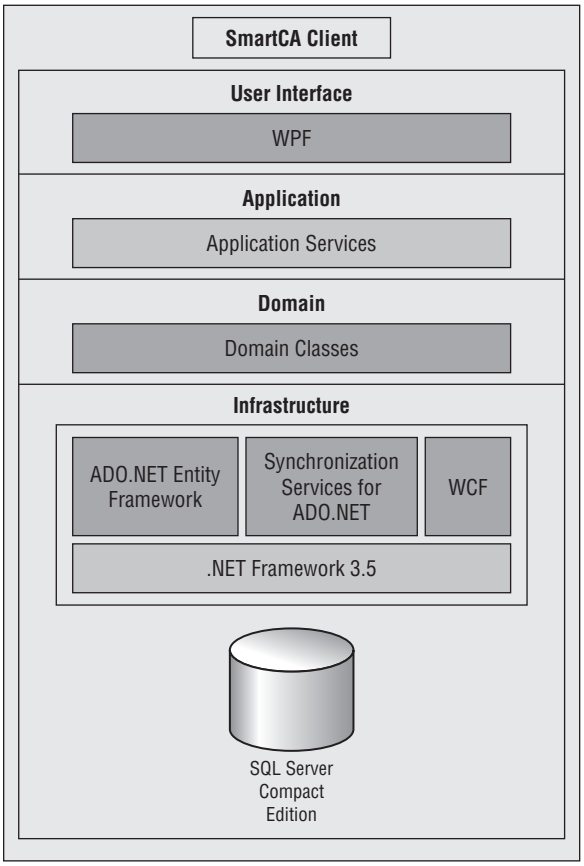

Figure 1.4: The SmartCA application architecture.

Fulfilling the Rich Client Application Functionality Requirement

Since the users of the current application have become used to a Windows application, the new application will also be Windows-based, but it will be much more than just a traditional Windows application. The SmartCA application will be a smart client application implemented using the Windows Presentation Foundation (WPF). You might be asking yourself, OK, what exactly do you mean by smart client?

A smart client is a type of application that combines the best of both Windows applications and web applications.

Windows Application Benefits

The advantages of Windows applications are that they are able to provide a rich user experience, they are not too complex to develop, and they can use local resources. Using local resources allows Windows applications to be responsive, interact with connected devices, and do other things that web applications cannot do (at least not very easily).

Web Application Benefits

The positive aspects of a web application are that it is easy to deploy and manage, since you deploy it to a server not to the client computer, and it has a very broad reach — even PDAs and cell phones can access a web application!

Smart Client Definition

The term "smart client" means different things to different people. For the purposes of this book, I will classify a smart client application (note this is adapted from the MSDN Smart Client FAQ) as follows:

- It uses local resources and provides a rich user experience. This satisfies the rich client application functionality requirement.
- It is a connected application that can exchange data on the Internet or on an enterprise network.
- Even though it is a connected application, it is offline capable so that it can be used even if it is not currently connected. This satisfies the Offline Capable requirement.
- It has an intelligent deployment and update story, maintaining relatively the same ease of deployment and management as web applications, thus satisfying the Intelligent Installation and Auto-Update Functionality requirement.

Intelligent deployment means that the smart client application is deployed to an application server, and from there it is deployed onto the local client system. Intelligent update means that the application on the client system is able to receive updates that are deployed to the server.

Windows Presentation Foundation (WPF)

WPF is intended to be the next-generation graphics API for Windows applications on the desktop. Applications written in WPF are visually of a higher quality than Windows Forms applications. Some of the relevant highlights of WPF for the SmartCA application are:

- **Resolution independence** — Because of WPF's use of vector graphics, unlike most Windows-based applications of today, graphics and text that are viewed in a higher resolution do not get smaller; they actually get better! This means that a user can literally shrink or enlarge elements on the screen independently of the screen's resolution.
- **Declarative programming** — Windows Forms do not have built-in support for declarative UI definitions. The .NET Framework as a whole has allowed developers to use declarative custom attributes classes, methods, and assemblies, as well as XML-based resource and configuration files. WPF takes this declarative-based model to a new level with Extensible Application Markup Language (XAML). Using XAML in WPF is very similar to using HTML to define a UI for a web page, yet it is much better than that analogy. Not only does XAML give a great range of expressiveness for the look and feel of a UI, but it also allows for parts of the behavior of the UI to be declarative.

- **Rich composition and customization** — It is very easy to customize controls in WPF with little or no code. Almost any type of control can be composed with another control. There literally are no boundaries here; for example, you could bind a media clip to a text box if you wanted to, or make it spin around, and so on. It is also very easy to "skin" applications with very different looks, without requiring any code. These advantages help satisfy the Rich Client Application Functionality requirement.
- **Easy deployment** — Using the .NET Framework's ClickOnce technology will provide a way to install and run SmartCA on the client machines simply by clicking on a URL. ClickOnce ensures that installation will not affect other applications because all files required for the application are placed in an isolated area and it also prevents any custom registration.

Fulfilling the Web Access Requirement

Although I have not talked much about the server-side partition of this application, the mere fact that there will be a server-side implementation means that it is possible for the application's data and behavior to be exposed to the web via a web application or even via web services. In fact, I will show later in the book how each SmartCA client application instance will be using web services to synchronize its transactions with the server.

Fulfilling the Intelligent Installation and Auto-Update Functionality Requirement

The SmartCA application will take advantage of the Visual Studio 2008's ClickOnce deployment tools and .NET Framework technology to satisfy the Intelligent Installation requirement. Since the .NET Framework also has built-in support for automatic updates and for rolling back to previous versions, the Auto-Update requirement will also be satisfied.

Since SQL CE is so lightweight (it only eats up about 1.5 MB worth of hard disk space), it will be very easy to deploy, which will also help support the Intelligent Installation requirement.

Summary

In this chapter, I introduced you to the fictitious company Smart Design, and more importantly, the new application that I am building for them, the SmartCA application. I also outlined the problems of the legacy application, the requirements for the new application, as well as what technologies and designs I plan to use to satisfy all of the requirements. In the next chapter, I will delve into the details of the layered architecture model and implementation for the SmartCA application.